Servos Unscrambled

Written By: Steven Robert Cypherd

SUMMARY

Servos bring life to our ideas. Servos can be mysterious, but in fact they are very simple. Servos use a simple electronic pulse to tell them what angle you want them to go to. It is electronic but not digital. There are digital servos and they are different than standard servos. Pulses are electronic, but they are part of our digital world. The abbreviation for microseconds is "us". The abbreviation for milliseconds is "ms". See pictures below.

I am using a standard servo as an example.

Servo inputs use pulses measured in microseconds (us) so that you have fine control over the angle you want the servo to go to. Microseconds are part of the language of microprocessors, not people. Functions that deal with microseconds do so in the way they were programmed.

People get scrambled up because functions like pulsout do not work the way they think they should. Remember they are made for the system they are running on. Read your manual. Most pulsout functions are called like this: pulsout pin, time. Time is the number of microseconds in each unit that will make up the pulse, not an actual time or angle. See pictures below.

Pulsout can be different on different processors in the same language because it is based on the clock speed of the processor. Also, it can be different in different versions of the same language. Read your manual. I think we should tell the companies to standardize functions like pulsout. It would make servos much more fun.

Servos from their Radio Control heritage need a stream of pulses about every 20

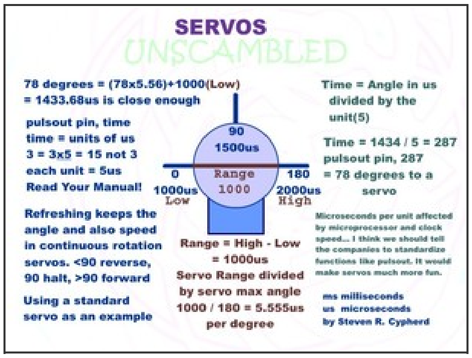
milliseconds or so for about ten times of the same pulse to get them to recognize that it is the angle you want them to go to. In your program just set up a loop to send a train of pulses to the servo when you first set it to each new angle. After that you only need to re-fresh you servo about every 30-50 milliseconds or as needed by your servo.

I make a constant of my time value like 28x2_TIME = 5. I state the maker, processor, speed and language version in a comment just so I know what I am using. This can be a nasty bug in a program to find.

For reading sensors like the Sharp rangefinders you should not use a servo command that constantly updates the servo position every 20 microseconds or so. Why? Because the sensor needs time to calculate the distance reading and to update its output. The servo needs to be steady to do this. Once you get good data from the sensor then you can move the servo. Try it. Use the pulsout command to move the servo to the next scan point. Then give about 20-50 microseconds pause for the sensor to update its output and then read the sensor. You should get good data this way. Digital and analogue rangefinders use the same method for reading distance and only differ in how you read the device.

See table below.

Step 1 — Servos Unscrambled



78 degrees = (78x5.56)+1000(Low)
= 1433.68us is close enough

pulsout pin, time
time = units of us
3 = 3x5 = 15 not 3
each unit = 5us
Read Your Manual!

Refreshing keeps the angle and also speed in continuous rotation servos. <90 reverse, 90 halt, >90 forward

Using a standard servo as an example

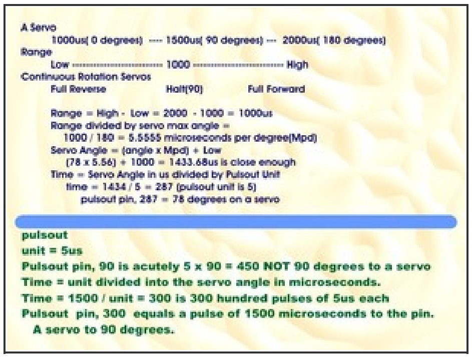
Range = High - Low
= 1000us
Servo Range divided by servo max angle
1000 / 180 = 5.555us per degree

Time = Angle in us divided by the unit(5)

Time = 1434 / 5 = 287
pulsout pin, 287
= 78 degrees to a servo

Microseconds per unit affected by microprocessor and clock speed... I think we should tell the companies to standardize functions like pulsout. It would make servos much more fun.

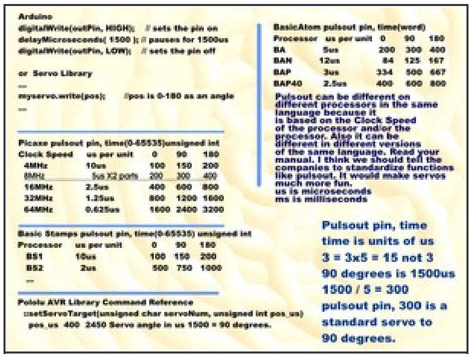
ms milliseconds
us microseconds
by Steven R. Cypherd



```
A Servo
        1000us( 0 degrees) ---- 1500us( 90 degrees) --- 2000us( 180 degrees)
Range
        Low ........................... 1000 ........................... High
Continuous Rotation Servos
        Full Reverse              Halt(90)          Full Forward

        Range = High - Low = 2000 - 1000 = 1000us
        Range divided by servo max angle =
           1000 / 180 = 5.5555 microseconds per degree(Mpd)
        Servo Angle = (angle x Mpd) + Low
            (78 x 5.56) + 1000 = 1433.68us is close enough
        Time = Servo Angle in us divided by Pulsout Unit
            time = 1434 / 5 = 287 (pulsout unit is 5)
                pulsout pin, 287 = 78 degrees on a servo
```

pulsout
unit = 5us
Pulsout pin, 90 is acutely 5 x 90 = 450 NOT 90 degrees to a servo
Time = unit divided into the servo angle in microseconds.
Time = 1500 / unit = 300 is 300 hundred pulses of 5us each
Pulsout pin, 300 equals a pulse of 1500 microseconds to the pin.
A servo to 90 degrees.



Arduino

```
digitalWrite(outPin, HIGH);   // sets the pin on
delayMicroseconds( 1500 ); // pauses for 1500us
digitalWrite(outPin, LOW);   // sets the pin off

or  Servo Library
---
myservo.write(pos);      //pos is 0-180 as an angle
---
```

Picaxe pulsout pin, time(0-65535)unsigned int

Clock Speed	us per unit	0	90	180
4MHz	10us	100	150	200
~~8MHz~~	~~5us X2 ports~~	~~200~~	~~300~~	~~400~~
16MHz	2.5us	400	600	800
32MHz	1.25us	800	1200	1600
64MHz	0.625us	1600	2400	3200

Basic Stamps pulsout pin, time(0-65535) unsigned int

Processor	us per unit	0	90	180
BS1	10us	100	150	200
BS2	2us	500	750	1000

Pololu AVR Library Command Reference
::setServoTarget(unsigned char servoNum, unsigned int pos_us)
pos_us 400 2450 Servo angle in us 1500 = 90 degrees.

BasicAtom pulsout pin, time(word)

Processor	us per unit	0	90	180
BA	5us	200	300	400
BAN	12us	84	125	167
BAP	3us	334	500	667
BAP40	2.5us	400	600	800

Pulsout can be different on different processors in the same language because it is based on the Clock Speed of the processor and/or the processor. Also it can be different in different versions of the same language. Read your manual. I think we should tell the companies to standardize functions like pulsout. It would make servos much more fun.
us is microseconds
ms is milliseconds

Pulsout pin, time
time is units of us
3 = 3x5 = 15 not 3
90 degrees is 1500us
1500 / 5 = 300
pulsout pin, 300 is a standard servo to 90 degrees.

- A servo using a standarc servo as an exampl.

Servos are fun!

This document was last generated on 2012-10-31 10:28:43 AM.